JITTER AND CLOCK RECOVERY FOR PERIODIC TRAFFIC IN BROADBAND PACKET NETWORKS

R. P. Singh, S. H. Lee and C. K. Kim

Bell Communications Research, Inc.
435 South st. Morristown NJ. 07960

ABSTRACT

The transparent transport of real-time periodic traffic through a broadband packet network requires the recovery of source clock frequency at the destination. The inherently stochastic nature of packet transport in the network makes the recovery difficult. In this paper, we illustrate the packet jitter phenomenon, describe several clock recovery algorithms, and provide a systematic design procedure for the implementation of these algorithms. We apply them to a computer model of a single stage FCFS multiplexer to demonstrate their effectiveness and different design trade-offs. We also discuss the application of the algorithms to the audio and video services.

1. INTRODUCTION

The transparent transport of real-time periodic traffic across a broadband packet network requires synchronization of the network. A simple way to achieve this would be to distribute clock timing throughout the network via a common master clock. Alternatively, a less expensive approach would be to run the clocks at each node autonomously with a nominal frequency, and make the destination clock frequency follow the source clock frequency as closely as possible. The source clock timing information can be explicitly carried in each packet header, as proposed for the narrowband integrated networks [3], for specific voice and video services on an end-to-end basis. The source clock timing information can also be derived from the arriving packet stream at the destination. This is achieved by monitoring the destination buffer level, which is a linear function of the frequency difference between the source and destination clocks, and using it to adjust the destination clock frequency without any buffer under- or overflow. These methods can also be combined together for some specific applications.

Due to the statistical nature of packet transport in a broadband packet network, a periodic packet stream of a real-time source traffic tends to become quasi-periodic. The destination buffer level, therefore, cannot be directly used as a measure of the frequency difference between the source and destination clocks due to random fluctuations, which may be correlated as well as nonstationary; its statistics may vary whenever there would be a change in the multiplexing and switching pattern due to addition or deletion of calls.

The principal questions in the design of synchronization and clock recovery schemes are: (i) how to separate the weak signal (the frequency difference between the source and the destination clocks) from the large noise (the packet jitter) to obtain the estimate of the former; and (ii) how to control the destination clock frequency using the frequency difference estimate. (The terms packet jitter and noise are used interchangeably in this paper.) The problem of synchronization and clock recovery in plesiochronous broadband packet networks has been discussed in [1,2] in rather general terms, with no specific design details. [4] addresses the problem in a quantitative manner. But their assumptions are quite restrictive, and the open loop control law suggested in [4] may lead to instability. In [5], a suboptimal and an optimal frequency difference estimation algorithms along with two control schemes for the adjustment of the destination clock frequency are described. These algorithms are based on realistic assumptions, and are different from those in [4].

In this paper, we focus our attention on the implementation of the algorithms of [5], and present a systematic design procedure. Section 2 illustrates the phenomenon of packet jitter via a simple example, where it is defined precisely in mathematical terms. Section 3 contains the estimation and control algorithms described in [5,6]. Implementation details are discussed in Section 4. Section 5 contains some simulation results by way of a computer model of a first come, first served (FCFS) multiplexer. Section 6 deals with service specific synchronization schemes, where the application of the above algorithms to audio and video services are discussed. Finally, Section 7 summarizes the results.

2. PACKET JITTER PHENOMENON

The phenomenon of packet jitter is illustrated in Fig. 1 for a single FCFS multiplexer with three packet streams of λ_1=1.0 packet per second (*pps*), λ_2=2.0 *pps* and λ_3=4.0 *pps* (packets are assumed to be of fixed length) which are multiplexed to a μ=8.0 *pps* line in the broadband network. The packets are removed synchronously at the destination without the destination buffer over- or underflow at roughly the same rate as that of the source buffer, e.g., λ_3 *pps*, as shown in the figure. Fig. 2a and 2b are the plots of the destination buffer level as a function of time, which show that its value is zero at the indicated sampling instants when there is no jitter introduced into the third packet stream, and that its value deviates from zero at the sampling instants when the packet stream has to contend for empty slots on the high speed line. (The positive slope in Fig. 2a is $(\mu-\lambda_3)$ and the negative slope is λ_3.)

Let f_1 be the source clock frequency in *pps*. (In the above example, $f_1=\lambda_3$.) Let $o(k)$ be the destination buffer level function. The packets are removed from the destination buffer at the destination clock frequency of $f_2(k)$ *pps*. Let $\Delta f(k)=f_1-f_2(k)$ be the difference between the source and destination clock frequencies. The rate of change of the destination buffer level indicates the measure of $\Delta f(k)$. When the packets in the input line arrive with stochastic delays, the input packet stream represents a noisy value of the source clock frequency, f_1, and $o(k)$ fluctuates in a random manner depending on $\Delta f(k)$ and the packet jitter characteristics. Let the noisy frequency be $f_1+n(k)$, where $n(k)$ is the random fluctuation in the source clock frequency due to the the stochastic inter packet spacing. Then the frequency difference between the incoming and the outgoing packet streams to and from the destination buffer is $f_1-f_2(k)+n(k)=\Delta f(k)+n(k)$. The dynamics of the state equation for $o(k)$ can now be derived from the conservation law. A convenient choice of the sampling interval is $T(k)=1/f_2(k)$, the time period of packet removal from the buffer.

$$o(k+1)=o(k)+T(k)(f_1-f_2(k)+n(k) \qquad (2.1)$$

or

$$\Delta\phi(k)=\phi(k+1)-\phi(k)=T(k)\Delta f(k)+d(k) \quad (2.2)$$

where $d(k)=n(k)T(k)$. $d(k)$ is the equivalent noise term in number of packets. The nonlinearity arising from the choice of $T(k)$ is ignored. (2.1) or (2.2) is the state equation of the buffer.

Definition of packet jitter: We define the random variable $d(k)$ as the packet jitter. When $\Delta f(k)=0$, the random change in $\phi(\cdot)$ from one time instant to another represents the packet jitter.

The most difficult part in the design and analysis of the clock recovery schemes is the modeling of the noise process, $d(k)$. Its expected value, of course, is zero, i.e, $Ed(k)=0$ ($E\cdot$ is the expectation operator.), by the fact that if the frequency difference is zero, then the average accumulation of packets in the destination buffer must be zero, assuming no packet loss through the network. The other statistics, in general, are not known, however. Some preliminary analysis indicates that $d(k)$ is expected to be a correlated sequence. The correlation is dependent on several factors - loading of the network; relative magnitudes of λ_i's in comparison to μ; service disciplines at the multiplexers and switches, e.g., FCFS or train scheduling [1]; type of connection, e.g., virtual circuit or datagram. These factors will influence the correlation of the packet jitter in a complex manner. However, in general the correlation in the jitter sequence results from a repeating pattern of jitter every few packets in the stream. If the packet stream goes through many stages of multiplexing and switching, the period of the pattern is large. This would mean less correlation. The plot of Fig. 2b shows this repeating pattern of packet jitter every 4 packets for the example considered above.

For simplicity, in this paper we do not explicitly consider the correlation of the process $d(k)$ in design. The jitter process, $d(k)$, is assumed to be stationary with $Ed(k)=0$ and an unknown constant variance, denoted by ν.

3. CLOCK RECOVERY ALGORITHMS

We assume that the network is plesiochronous, i.e., the clocks at the source and the destination derive their timings from independent local clocks, with the same nominal frequency but different *ppm* values. The frequency difference between the two end clocks, $\Delta f(k)$, is quite small - typically a few tenth of a *ppm*. But the values of $d(k)$ may vary from 0 packet to a few packets from one instant to another. Thus, the changes in $o(k)$ are mainly due to the random jitter $d(k)$. However, since $Ed(k)=0$, we expect that the contribution of the $\Delta f(k)$ term to the buffer level function in $j(\cdot)T(k)$ time periods may become quite significant, where $j(\cdot)$ denotes the length of the interval, which is large and time-varying. Thus, it may be possible to estimate the frequency difference from the data set $o(k)$, $k\geq 0$ if we wait long enough, i.e., for long enough $j(\cdot)$.

We propose the following two time-scale model - a fast time sequence, $k\geq 0$, when we collect observation data $o(k)$, and a slow time sequence, $m\geq 0$, on which we compute the estimates of the frequency difference, denoted by $\Delta f(m)$, from the set $o(k)$, $k\geq 0$, and adjust $f_2(m)$ using this estimated frequency difference. The slow time scale interval $[m, m+1)$ is made time-varying, i.e., $0\leq k\leq j(m)$ for the slow time scale interval. We propose to use a time-varying time period governed by

$$j(m+1)=\frac{1}{(1-\alpha)}j(m) \quad (3.1)$$

where $j(m)$ is the length of the interval $[m, m+1)$, with $j(0)$ as its initial value. $(1-\alpha)$ is a design parameter which controls the rate of decay of $\Delta f(m)$ as described later. The size of the interval increases at the same rate at which $\Delta f(m)$ decreases. The idea behind using this scheme is that in the beginning when the frequency difference is large, we want more frequent control actions, perhaps at the cost of poor estimate. As time progresses, the frequency difference diminishes exponentially at $(1-\alpha)$ rate, but the noise acts persistently, resulting in a decreasing signal to noise ratio with time. Therefore, we must have a larger interval for a better estimate of the decreasing $\Delta f(m)$ with m.

We process the observation set $\phi(k)$, $0\leq k\leq j(m)$ in some manner to filter out the effects of the rapidly changing (from packet to packet) noise and obtain the frequency difference estimate $\Delta f(m)$ for each interval $[m, m+1)$, which is used in the control algorithms below to compute $f_2(m+1)$. Note that $\Delta f(m)$ is constant in each interval $[m, m+1)$. Let $\Delta f(m)$ denote the frequency difference estimation error, i.e., $\Delta f(m)=\Delta f(m)-\Delta f(m)$. The estimation algorithms described later are such that the estimate $\Delta f(m)$ is unbiased, i.e., $E\Delta f(m)=E\Delta f(m)$. Further, if $d(k)$ is a white noise sequence, then the resulting $\Delta f(m)$ is also a white noise sequence.

3.1 Frequency Adjustment Algorithm

Before we describe the algorithms to estimate the frequency difference, let us discuss the control algorithms. Since $\Delta f(m)$ is not known, its estimate $\Delta f(m)$ is used as an input to the control algorithms.

Control Law I: A simple control law is to add a weighted value of $\Delta f(m)$ to $f_2(m)$ to obtain $f_2(m+1)$, which results in a first order dynamic system, i.e.,

$$f_2(m+1)=f_2(m)+\alpha\Delta f(m) \quad (3.2)$$

for some $0<\alpha<1$. With the above definition of $\Delta f(m)$, the closed loop system is

$$f_2(m+1)=(1-\alpha)f_2(m)+\alpha(f_1+\Delta f(m))$$

which is stable for the choice of α. Since $Ef_1=f_1$ and $E\Delta f(m)=0$, the mean of $f_2(m)$ satisfies

$$Ef_2(m+1)=(1-\alpha)Ef_2(m)+\alpha f_1 \quad (3.3)$$

whose asymptotic value can be shown to be $Ef_2(\infty)=f_1$.

The dynamics of the variance $\pi(m)=E(f_2(m)-Ef_2(m))^2$ can be derived from (3.2) assuming that $\Delta f(m)$ is a white noise sequence, and is given by

$$\pi(m+1)=(1-\alpha)^2\pi(m)+\alpha^2\sigma(m) \quad (3.4)$$

where $\sigma(m)=E\Delta f^2(m)$ is the variance of the frequency difference estimation error. Since $0<(1-\alpha)<1$, the asymptotic variance (as $m\to\infty$) of $f_2(m)$ can be shown to be $\pi(\infty)=\frac{\alpha}{2-\alpha}\sigma(\infty)$. $\sigma(\infty)$ is related to ν through the estimation algorithms described later. The residual variance $\pi(\infty)$ is smaller than $\sigma(\infty)$ for all $0<\alpha<1$. There is a trade-off between how fast the dynamic response should be and how much variance reduction we can obtain. A larger α means faster response ($(1-\alpha)$ is closer to zero), but higher residual variance, and vice versa. Also, a larger α implies a smaller destination buffer.

Control Law II: The control algorithm, in which the value of $f_2(m+1)$ is obtained by adding the weighted values of $f_2(m)$, $f_2(m-1)$, $\Delta f(m)$ and $\Delta f(m-1)$, provides more freedom in controlling the rate of convergence and the asymptotic variance. The weightings on $f_2(m)$ and $f_2(m-1)$ should be such that the open loop transfer function should contain $(1-z^{-1})$ as a factor in the denominator, so that there is no steady state error. (z is a forward shift operator, i.e., $zx(t)=x(t+1)$.) The new control algorithm results in a second order dynamic system

$$f_2(m+1)=(1+\delta)f_2(m)-\delta f_2(m-1)+\alpha\Delta f(m)+\beta\Delta f(m-1)$$

where δ, α and β are free parameters, which can be determined on the basis of the choice of the two closed loop poles, while minimizing $\pi(\infty)$. See [5,6] and the references therein for detail.

3.2 Frequency Estimation Algorithms

We describe below two increasingly complex algorithms for

computing $\Delta f(m)$. Both algorithms provide estimates of $\Delta f(m)$ by processing the finite data set $\Delta\phi(k)$, $0 \le k \le j(m)$ in the interval $[m, m+1)$. For convenience, the buffer state equation (2.2) is rewritten here as

$$\Delta\phi(k) = \frac{\Delta f(m)}{f_2(m)} + d(k) \tag{3.5}$$

Simple Time-Averaging Estimation Algorithm: An ad hoc scheme to estimating $\Delta f(m)$ may be just to take the sample mean of all the available observations $\Delta\phi(\cdot) f_2(m)$. This is motivated by the idea that the zero mean noise samples will cancel out by averaging it. There are a total of $j(m)$ samples in the interval $[m, m+1)$, and thus,

$$\Delta f(m) = \frac{1}{j(m)} \sum_{k=0}^{j(m)-1} \Delta\phi(k) f_2(m)$$

$$= \frac{f_2(m)}{j(m)} (\phi^-(m+1) - \phi(m)) \tag{3.6}$$

where $\phi^-(m+1)$ is the value of $\phi(\cdot)$ just before the control action is taken at the instant $m+1$.

Optimal Estimation Algorithm: The optimal estimation algorithm makes use of $\pi(0)$ and ν, and is based on the first control law. The Appendix contains a mechanism by which the variance of the noise is estimated in real time using the same set of data $\Delta\phi(\cdot)$. The following set of equations for the optimal estimation algorithm are developed in [5,6] from the basic Kalman filter equations and the first control law

$$\Delta f(m) = \frac{p(m)}{1+j(m)p(m)} (\phi^-(m+1) - \phi(m)) f_2(mm$$

$$+ \frac{1}{1+j(m)p(m)} E\Delta f(m) \tag{3.7}$$

$$p(m+1) = (1-\alpha)^2 p(m) + \alpha^2 \frac{p(m)}{1+j(m)p(m)} \tag{3.8}$$

with $p(0) = \frac{\pi(0)}{f_2^2(0)\nu}$ known. $\phi^-(m+1)$ is the value of the buffer level function at the instant just before $m+1$. $E\Delta f(m)$, $j(m)$ and $f_2(m)$ are all known quantities in $[m, m+1)$. For the purpose of computation, we approximate $f_2(m) = f_2(0)$. Since ν is not known, we propose to use the variance estimation algorithm, (A.1) or (A.2), to estimate it in the interval [0,1) (assuming stationarity), which will be used during the other intervals as well. (3.9) is obtained from the quadratic Riccati equation of the Kalman filter equations (see [5,6] for detail.).

In both cases, the algorithms result in finite estimates of frequency difference for each interval, and $\sigma(m) \to 0$ as $j(m) \to \infty$ with $m \to \infty$ if $d(k)$ is assumed to be a white noise sequence. The first algorithm is applicable to all systems regardless of whether $d(k)$ is correlated or not. The second algorithm may give poor performance if the sequence $d(k)$ is highly correlated. As said earlier, modeling of correlation of $d(k)$ is difficult. We are continuing our efforts to understand the various contention mechanisms which cause correlation in packet jitter. Even then, it does not appear straight forward to develop optimal estimation algorithms for the correlated $d(k)$ case.

Fig. 3 shows the signal flow diagram for the estimation and control algorithms along with the two time scale model.

4. IMPLEMENTATION OF THE ALGORITHMS

Let $f_1 = f \pm f\epsilon_s$, where f is the nominal frequency and ϵ_s is its standard deviation in *ppm*, i.e., $f - \frac{f\epsilon_s}{10^6} \le f_1 \le f + \frac{f\epsilon_s}{10^6}$. Likewise, let $f - \frac{f\epsilon_d}{10^6} \le f_2(0) \le f + \frac{f\epsilon_d}{10^6}$. This suggests that $Ef_1 = Ef_2(0) = f$ and the variance of f_1 is $\frac{f^2\epsilon_s^2}{10^{12}}$ and that of $f_2(0)$ is $\frac{f^2\epsilon_d^2}{10^{12}}$. Therefore, $E\Delta f(0) = 0$ and $\pi(0) = \frac{f^2(\epsilon_s^2+\epsilon_d^2)}{10^{12}}$ should be used in the optimal estimation algorithm, once it is ascertained that $d(k)$ is close to being a white noise sequence.

Now, the next question is how to choose $j(0)$? $j(0)$ is determined from the prior knowledge of $\Delta f(0)$ (or $\pi(0)$), the maximum value of the noise magnitude and the allowable buffer size, so that even if the system is running for $j(0)$ units of time in the presence of noise, there is no buffer under- or overflow, and we get a reasonable estimate of $\Delta f(0)$ at the end of the first interval. In practice, after a few iterations $\Delta f(m)$ will become very small. At this time we can stop changing $j(m)$ and use a constant value. We propose that for some $m = m_1$, whenever either $|\Delta f(m_1)| < \epsilon_1$, $\pi(m_1) < \epsilon_2$, or $\frac{\pi(m_1)}{\nu(m_1)} < \epsilon_3$, then set $j(m) = j(m+1)$ for all $m \ge m_1$ for the steady state operation. Where ϵ_1, ϵ_2 and ϵ_3 are design parameters.

In the above algorithms, the time intervals $[m, m+1)$, $m \ge 0$ are determined a priori, and the destination buffer levels are read at time instants, $m \ge 0$. The values of these may contain a fraction of packets. Alternatively, it is also possible to fix the destination buffer level function at discrete packet levels, e.g., half, three quarter etc., and to measure the time intervals for $\phi(k)$ to attain these predetermined labeled values. There are two limitations of this technique: (i) $\phi(k)$ may occasionally cross the labeled points due to a large but unexpected jitter. If for some reason these crossings occur too frequently, the estimation of the frequency difference may be quite poor due to small number of observations, which will result in a larger residual jitter. (ii) By observing $\phi(k)$ at a coarse level, we loose information about the statistics of the high frequency jitter component buried in it. Thus, the variance of the jitter cannot be estimated on-line from this observation set, and the Kalman algorithm cannot be used.

5. SIMULATION RESULTS

We used a computer model of a FCFS multiplexer to study the estimation and control algorithms of Section 2. The inputs to the multiplexer consist of a combination of signals, including DS0 and DS3. The output rate of the multiplexer is 150 Mbps. We implemented both estimation algorithms ((3.6) as well as (3.7) and (3.8)) in the first control algorithm (3.2) with the parameter α varying between 0 and 1 to illustrate the trade off between the speed of convergence and the residual variance. The algorithms are implemented to synchronize clocks at DS0 and DS3 rates - the two extremes of the DS-hierarchy. The various values used in the simulations are: For DS0 signal - $f_1 = 64.00$ kbps, $f_2(0) = 63.99$ kbps, and $\Delta f(0) = 156$ *ppm*; and for DS3 signal - $f_1 = 44736.00$ kbps, $f_2(0) = 44735.11$ kbps, and $\Delta f(0) = 20$ *ppm*.

The choice of DS0 and DS3 signals was dictated by the expectation that they generate jitter with different degrees of correlatedness. We first compute the autocorrelation function of the jitter generated by these two signals. Figs. 4 and 6, respectively, show the plots of normalized autocorrelation function (normalized by variance) for the DS0 and DS3 signals. We observe that the magnitude of the jitter generated by the DS0 signal is small (standard deviation = 0.006 packets) with a decaying correlation. (See Fig. 4.) The signal to noise ratio is $\frac{0.01}{64 * 0.006} = 0.026$. On the other hand, the DS3 signal provides a large jitter (standard deviation = 0.26 packets) with a large and persistent correlation and a shorter period of jitter pattern. (See Fig. 6.) The signal to noise ratio in this case is $\frac{0.89}{44736 * 0.26} = 0.00008$. which is much smaller than before.

As expected, therefore, the estimation algorithm (3.7) with the control law (3.2) results in a superior performance (see Fig. 5a) in terms of reduced jitter for the DS0 signal as compared to the ad hoc scheme (3.6) with (3.2) (see Fig. 5b). The value of $\alpha = 0.5$. Solid lines denote the estimated

frequencies and dashed lines represent the buffer levels.

Figs. 7a, 7b and 7c show the plot of the first estimation (3.6) and control algorithm (3.2) for DS3 signal with different values of α ; $\alpha \approx$ 0.25, 0.50, and 0.75. In these plots, we observe the effect of the parameter α on the convergence speed and the rapid fluctuation in $\Delta f(m)$. With a larger α, the estimated frequencies converge to the source frequencies faster but show greater fluctuations. However, because the estimation and control algorithms correct the frequency difference quickly when α is large, the system operates at smaller buffer levels despite the variations in the estimated frequency. (See Fig. 8.) Since the packet jitter is highly correlated for the DS3 signal, the optimal estimation algorithm results in very poor estimates, and consequently poor convergence.

6. SERVICE SYNCHRONIZATION

The clock recovery schemes presented earlier do not utilize the characteristics of individual services such as voice or video. Moreover, they are sensitive to the packet loss and large stochastic network delays. Large network delays mean a larger fixed delay for the original signal. A loss of packet means that the incoming packet stream is not a true measure of the source clock frequency. Thus, it appears that the potential application of the above clock recovery schemes will be limited to the packet transport of the existing digital circuit-switched hierarchical trunk signals. For an end-to-end packet-switched network, one can employ more flexible service-dependent clock recovery schemes, utilizing the properties of the individual services.

6.1 Voice Signal

The end-to-end voice synchronization scheme presented here is different than the ones in the literature [3]. For the synchronization of a voice signal, it is not necessary to use the clock recovery schemes of the previous section to adjust the destination clock frequency. The destination buffer under- and overflow due to the frequency difference between the source and the destination clocks can be controlled by occasionally deleting or repeating packets in the voice packet stream at the receiving end after delaying the packet stream by the maximum jitter amount. For example, with a clock frequency difference of 10 *ppm*, the average service interruption interval (average period of packet deletion or addition) would be more than 26 minutes with a 1000 bit packet length for 64 kbps voice call in a 150 Mbps packet network. Moreover, in practice, this clock difference can be corrected by deleting or adding packets during the silence interval without affecting a normal telephone conversation.

6.2 Video Synchronization

Depending on the video signal rate, type of coding and the desired quality of the video service, one may use any of the following methods for synchronization of end-to-end terminals.

6.2.1 Without Clock Recovery Schemes:

Similar to the previous case, independent local clocks with the same nominal frequency can be used to achieve synchronization between a video encoder at the source and a decoder at the destination. The incoming packets can be delayed by a given amount of time (more than the maximum tolerable jitter, possibly one video frame delay) and played out according to the local decoding clock. Terminal synchronization can be achieved by repeating or deleting an entire video frame at the destination buffer, depending on the buffer level. This is like the phenomenon of slip in the digital circuit switch interface. If the intra-frame coding is used, then packets corresponding to one whole frame can be deleted or repeated prior to decoding. This can be accomplished by inserting special frame sync pattern in the frame boundary at the transmitter to distinguish one frame from the other. If the clock difference is 1 *ppm*, then frame repetition or deletion will happen once in about every 9 hour interval, which is acceptable for most of the video services.

6.2.2. With Clock Recovery Schemes: *Fixed Bit Rate Video:*

Straightforward PCM or fixed length DPCM type of simple coding schemes generate a constant bit rate, and packetization of this coded bit stream results in a periodic packet stream at the source. The clock recovery scheme described in the previous section can be effectively used to synchronize clocks at the two ends, assuming a reasonable jitter and no packet loss.

Variable Bit Rate Video: Variable bit rate generation is inevitable for almost all video coding algorithms that utilize local image statistics, except for straightforward PCM or some low compression coding. This coding scheme seems to be a natural choice for a packet-switched network. In this case, however, the clock recovery schemes of the previous section cannot be used directly with the variable rate packet stream, since it is difficult to distinguish the variation in the periodicity of the packet stream due to the statistical coding and due to the stochastic network delays. Therefore, the source has to provide an extra timing information to the destination, so that the clock recovery circuit can lock on to that timing information. The timing information can include sequence number (video line or frame number) and a special pattern to easily detect the sequence. This timing information can be inserted into the variable bit stream prior to packetization. If the packet length is short, it is also possible to insert separate timing packets into the packet stream, and the source clock frequency can be recovered from this jittered timing packets based on the schemes described in the previous section.

Subband Coding: In the subband coding scheme an input video signal is decomposed into several different frequency bands, and different coding schemes are applied in each frequency band to take advantage of the local image statistics. This generates multiple parallel bit streams. The idea behind the clock recovery for such video signals is to apply a fixed bit rate coding scheme in one band, possibly the lowest frequency band, and recover the source clock frequency from the packet stream coming from that band at the destination.

7. CONCLUDING REMARKS

The difficulty in clock recovery for real-time periodic traffic in a broadband packet network stems from the random nature of packet transport through the network. We have presented several schemes by which the stochastic effects on the periodic packet stream can be mitigated, and the source clock frequency can be recovered asymptotically. We have discussed the application of these schemes to the audio and video services, as well as to the digital circuit-switched traffic, e.g., DS1, DS3. A systematic implementation procedure has been presented, and the effectiveness of the schemes have been evaluated via simulations by way of a computer model of a single stage FCFS multiplexer for different source traffic rates.

REFERENCES

[1] S. H. Lee, "An Integrated Transport Technique for Circuit and Packet Switched Traffic," IEEE INFOCOM '88, New Orleans, LA, March 1988.

[2] J. Y. Cochennec, P. Adam and T. Houdoin, "Asynchronous Time Division Networks: Terminal Synchronization for video and sound signals," IEEE GLOBECOM'85, pp 791-794, November 1985.

[3] W. A. Montgomery, "Techniques for Packet Voice Synchronization," *IEEE Journal on Selected Areas in Communications*, Vol SAC-1, No. 6, pp 1022-1028, December 1983.

[4] M. De Prycker, et al, "Terminal Synchronization in Asynchronous Networks, International Communications Conference, Seattle, WA, pp 22.7.1-8, June 1987.

[5] R. P. Singh and S. H. Lee, "Adaptive Clock Synchronization Schemes for Real-Time Traffic in Broadband Packet Networks," 8th European Conference on Electrotechnics, Stockholm, Sweden, June 1988.

[6] R. P. Singh, S. H. Lee and C. K. Kim, "Adaptive Clock Synchronization Schemes for for Real-Time Traffic in Broadband Packet Networks," to be submitted to *IEEE Transactions on Communication*, Sept. 1988.

APPENDIX

On-line Computation of Noise Variance Estimate, $\hat{v}(\cdot)$: $v(k)$ can be estimated from the available information, $\Delta\phi(k+1)$, $\Delta\phi(k)$, ..., $\Delta\phi(0)$ as follows.

Approximate Analysis: Since $\Delta f(k)$ is small, we can ignore its effect on $\Delta\phi(k)$ on the fast time scale and write $\Delta\phi(k) = d(k)$ for all $k \geq 0$. Hence, $v = Ed^2(k) = E\Delta\phi^2(k)$ Now, the variance, v, can be estimated as follows ($\hat{v}(k)$ is the estimate of v at time k.)

$$\hat{v}(k+1) = \frac{k}{k+1}\hat{v}(k) + \frac{1}{k+1}\Delta\phi^2(k) \qquad \text{(A.1)}$$

for all $\infty > k \geq 0$, with $\hat{v}(0) = \Delta\phi^2(0)$, or some initial guess of $\hat{v}(0)$, if it is available.

Exact Analysis: Successive differencing of the observation $\Delta\phi(\cdot)$ yields $\Delta\Delta\phi(k) = \Delta\phi(k+1) - \Delta\phi(k) = d(k+1) - d(k)$. The expected value of $\Delta\Delta\phi(k)$ is zero, and its variance is given by $E\Delta\Delta\phi^2(k) = E(d(k+1) - d(k))^2$. Since the noise is uncorrelated and stationary, this yields $Ed^2(k+1) + Ed^2(k) = E\Delta\Delta\phi^2(k) = 2v$. Analogous to (A.12), the variance estimate, $\hat{v}$, can be computed via the following recursion

$$\hat{v}(k+1) = \frac{k}{k+1}\hat{v}(k) + \frac{1}{k+1}\frac{1}{2}\Delta\Delta\phi^2(k) \qquad \text{(A.2)}$$

with $\hat{v}(0) = \frac{\Delta\Delta\phi^2(0)}{2}$ as the initial condition.

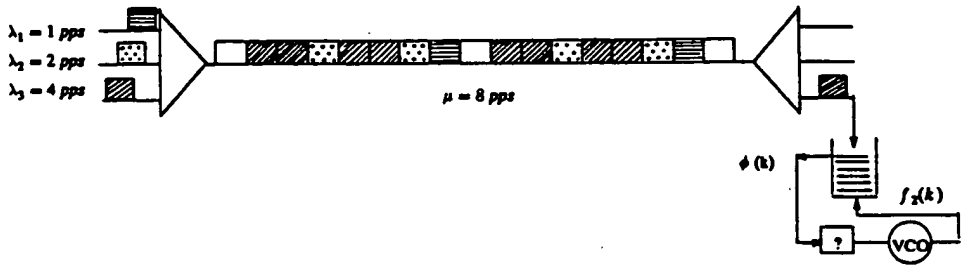

Figure 1 : A Network with a FCFS Multiplexer

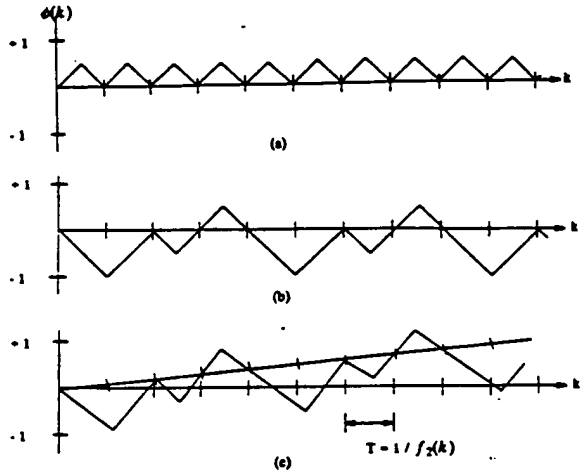

Figure 2 : Buffer Filling Level as a Function of Time

(a) No contention and $\Delta f = 0$

(b) Contention and $\Delta f = 0$

(c) Contention and $\Delta f \neq 0$

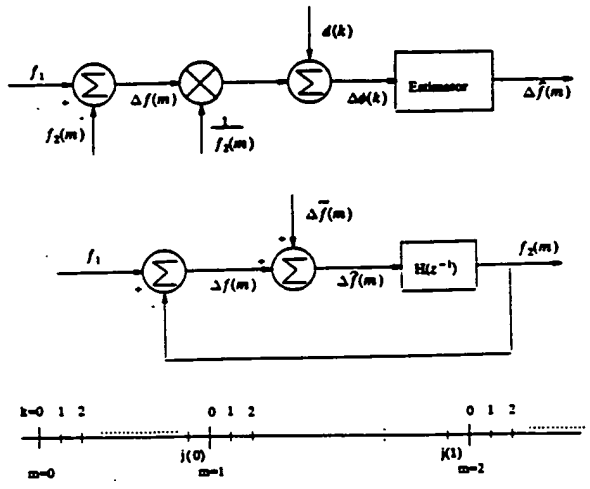

Figure 3 : Signal Flow Diagram for Estimation and Control Algorithms on a Two Time Scale

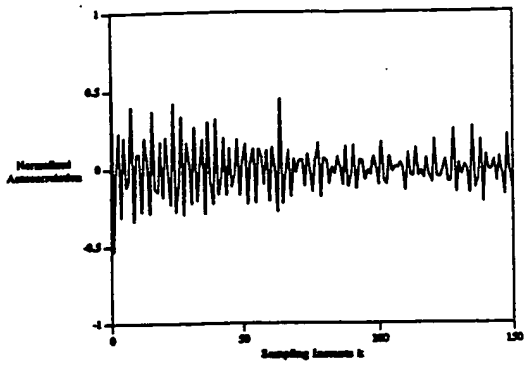

Figure 4 : Plot of Normalized Jitter Autocorrelation Function
(64 kbps)

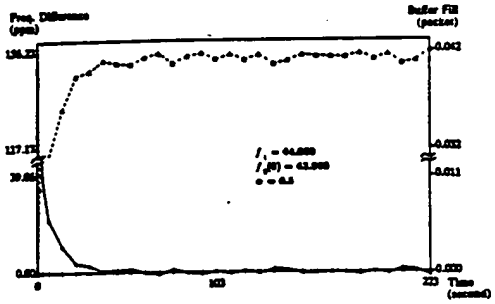

Figure 5a : $f_1(m)$ and $\phi(m)$ versus m
Control Algorithm I, Estimation Algorithm I

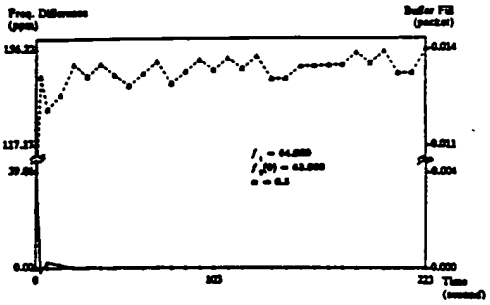

Figure 5b : $f_1(m)$ and $\phi(m)$ versus m
Control Algorithm I, Estimation Algorithm II

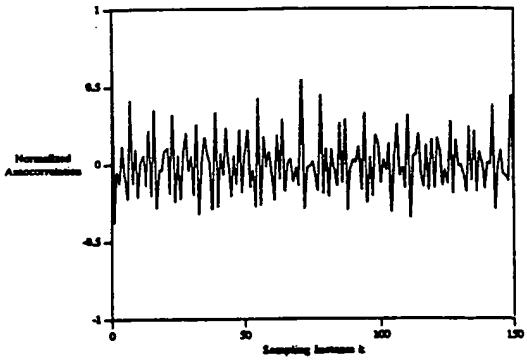

Figure 6 : Plot of Normalized Jitter Autocorrelation Function
(44736 kbps)

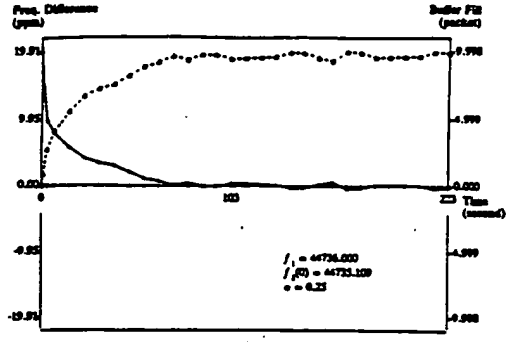

Figure 7a : $f_1(m)$ and $\phi(m)$ versus m
Control Algorithm I, Estimation Algorithm I

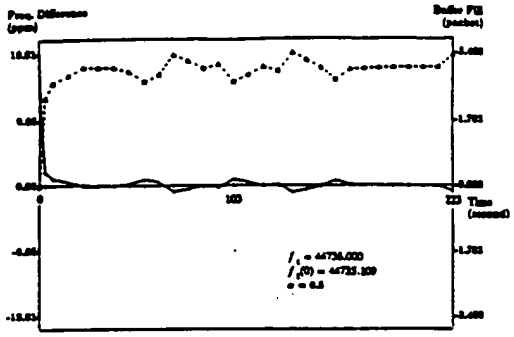

Figure 7b : $f_1(m)$ and $\phi(m)$ versus m
Control Algorithm I, Estimation Algorithm I

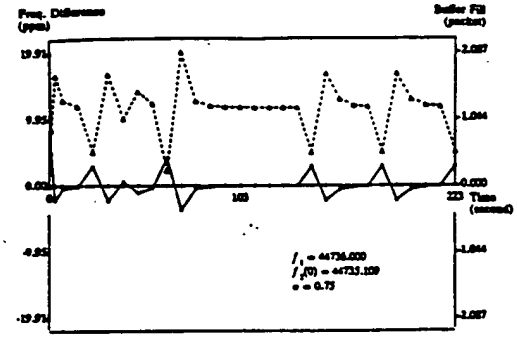

Figure 7c : $f_1(m)$ and $\phi(m)$ versus m
Control Algorithm I, Estimation Algorithm I

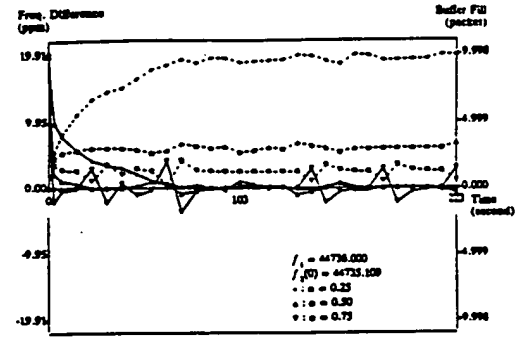

Figure 8 : $f_1(m)$ and $\phi(m)$ versus m
Control Algorithm I, Estimation Algorithm I